IDA PAPER P-1869

WIS-Ada FOUNDATION TECHNOLOGY PROGRAM PRELIMINARY PROTOTYPE SPECIFICATION

John Salasin
Murray Berkowitz
Mike Bloom
Bill Brykczynski
Joseph Hrycyszyn
Vance Mall

July 1985

Prepared for

WIS Joint Program Management Office

INSTITUTE FOR DEFENSE ANALYSES
1801 N. Beauregard Street, Alexandria, VA 22311

IDA Log No. HQ 85-30272

DEFINITIONS

IDA publishes the following documents to report the results of its work.

Reports

Reports are the most authoritative and most carefully considered products IDA publishes. They normally embody results of major projects which (a) have a direct bearing on decisions affecting major programs, or (b) address issues of significant concern to the Executive Branch, the Congress and/or the public, or (c) address issues that have significant economic implications. IDA Reports are reviewed by outside panels of experts to ensure their high quality and relevance to the problems studied, and they are released by the President of IDA.

Papers

Papers normally address relatively restricted technical or policy issues. They communicate the results of special analyses, interim reports or phases of a task, ad hoc or quick reaction work. Papers are reviewed to ensure that they meet standards similar to those expected of refereed papers in professional journals.

Documents

IDA Documents are used for the convenience of the sponsors or the analysts to record substantive work done in quick reaction studies and major interactive technical support activities; to make available preliminary and tentative results of analyses or of working group and panel activities; to forward information that is essentially unanalyzed and unevaluated; or to make a record of conferences, meetings, or briefings, or of data developed in the course of an investigation. Review of Documents is suited to their content and intended use.

The results of IDA work are also conveyed by briefings and informal memoranda to sponsors and others designated by the sponsors, when appropriate.

The work reported in this document was conducted under contract MDA 903 84 C 0031 for the Department of Defense. The publication of this IDA document does not indicate endorsement by the Department of Defense, nor should the contents be construed as reflecting the official position of that agency.

This paper has been reviewed by IDA to assure that it meets high standards of thoroughness, objectivity, and sound analytical methodology and that the conclusions stem from the methodology.

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION Unclassified	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release, unlimited distribution.
2b DECLASSIFICATION/DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S) IDA Paper P-1869	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION Institute for Defense Analyses 6b OFFICE SYMBOL IDA	7a NAME OF MONITORING ORGANIZATION OUSDA, DIMO
6c ADDRESS (City, State, and Zip Code) 1801 N. Beauregard St. Alexandria, VA 22311	7b ADDRESS (City, State, and Zip Code) 1801 N. Beauregard St. Alexandria, VA 22311
8a NAME OF FUNDING/SPONSORING ORGANIZATION WIS Joint Program Management Office 8b OFFICE SYMBOL (if applicable) WJPMO	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER MDA 903 84 C 0031
8c ADDRESS (City, State, and Zip Code) WIS JPMO/DXP Room 5B19, The Pentagon Washington, D.C. 20330-6600	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. T-W5-206 / WORK UNIT ACCESSION NO.
11 TITLE (Include Security Classification) WIS-Ada Foundation Technology Program. Preliminary Prototype Specification (U)	
12 PERSONAL AUTHOR(S) John Salasin, Murray Berkowitz, Michael I. Bloom, Bill R. Brykczynski, Joseph Hrycyszyn, Vance Mall	
13a TYPE OF REPORT Final 13b TIME COVERED FROM _____ TO _____	14 DATE OF REPORT (Year, Month, Day) 1985 July 15 PAGE COUNT 128
16 SUPPLEMENTARY NOTATION	
17 COSATI CODES FIELD / GROUP / SUB-GROUP	18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number) Ada programming language; prototyping; foundation technology; World Wide Military Command and Control System (WWMCCS) Information System (WIS); (Continued)

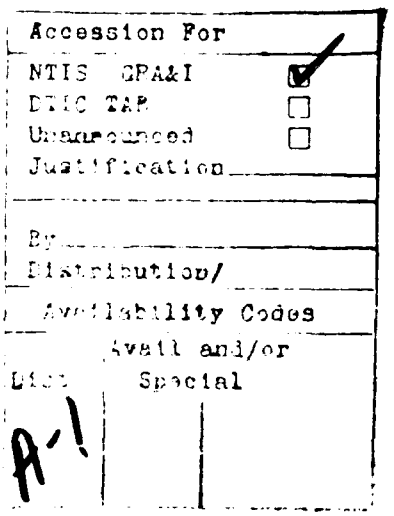
19 ABSTRACT (Continue on reverse if necessary and identify by block number)

The purpose of IDA Document P-1869, *WIS-Ada Foundation Technology Program. Preliminary Prototype Specification*, is to describe the general requirements for developing prototypes which will produce software components. These software components will perform demonstrations of the functionality required by the World Wide Military Command and Control System (WWMCCS) Information System (WIS), that is, (1) use the Ada programming language to provide portability, reliability, and maintainability; and (2) maintain consistency with current and "in process" software standards. Foundation areas in which prototypes will be developed include command languages, software design, text processing, database management systems, graphics, and network protocols.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION Unclassified
22a NAME OF RESPONSIBLE INDIVIDUAL Mr. Bill R. Brykczynski	22b TELEPHONE (Include area code) (703) 824-5515 22c OFFICE SYMBOL IDA/CSED

18 SUBJECT TERMS (Continued)

portability; reliability; maintainability; command languages; software design; text processing; database management systems; graphics; network protocols.



IDA PAPER P-1869

WIS-Ada FOUNDATION TECHNOLOGY PROGRAM PRELIMINARY PROTOTYPE SPECIFICATION

John Salasin
Murray Berkowitz
Mike Bloom
Bill Brykczynski
Joseph Hrycyszyn
Vance Mall

July 1985

INSTITUTE FOR DEFENSE ANALYSES

Contract MDA 903 84 C 0031
Task T-4-206

Acknowledgments

The authors are indebted to the consultants for their invaluable contributions to this paper. A list of the consultants appears in Appendix A of this document.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

1.0 INTRODUCTION

Several projects are planned to prototype foundation technologies for the World Wide Military Command and Control System (WWMCCS) Information System (WIS) using the Ada programming language. The purpose for developing these prototypes is to produce software components that perform the following:

a. Demonstrate the functionality required by WIS.

b. Use the Ada programming language to provide maximum possible portability, reliability, and maintainability consistent with efficient operation.

c. Maintain consistency with current and "in-process" software standards.

Foundation areas in which prototypes will be developed include the following:

a. Command language(s)

b. Software design and analysis tools

c. Text processing

d. Database tools

e. Operating systems

f. Planning and optimization tools

g. Graphics

h. Network protocols

This report describes general requirements for prototype projects and outlines potential efforts in each of the eight foundation areas.

2.0 GENERAL REQUIREMENTS FOR WIS PROTOTYPE PROJECTS

The design and development of Ada software prototypes will satisfy each of the following general requirements in the areas of software development methodology, design approach, language/implementation features, and administrative reporting.

2.1 Development Methodology

A rapid prototyping approach to software development is encouraged. Each effort should plan and produce a sequence of prototype executable Ada modules demonstrating increasing

functionality and capability. Ada will be used as a Program Design Language (PDL).

2.2 Design Approach

2.2.1 Software Engineering

Projects will employ software engineering techniques that perform the following tasks:

a. Maximize the reuse of existing design and/or code (available components should be identified).

b. Assure a high degree of modularity consistent with requirements for modifiability and maintainability.

c. Minimize, identify, and segregate all hardware, environment, and interface dependent functions.

d. Regulate programming style (including design style, presentation style, and the style of applying the language).

2.2.2 Program Design Language (PDL)

An Ada PDL will be used as an integral part of the design and development process. The development methodology will be based on incremental refinement and functional expansion of this PDL. It is anticipated that most efforts will use the WIS Standard PDL.

2.2.3 Security

Components will, where required, be designed to meet the National Security Agency's (NSA) Security Center's Guidelines for B3 Classification. While several foundation technologies (e.g. planning and optimization tools) may not require the incorporation of security guidelines in their design, others (e.g., operating system prototypes) must be constructed in accordance with these guidelines.

2.3 Language/Implementation

All software will be constructed to ensure the following:

a. Proper language considerations with respect to reusability:

(1) Coherent encapsulation of logically related entities and computational resources within packages

(2) Maximum use of generic program units to factor out common algorithms

(3) Minimum use of implementation-dependent features of the Ada programming language

b. Appropriate use of the Ada tasking model for expressing concurrent activities

2.4 Reporting

These prototype efforts are planned to avoid onerous and expensive reporting. Reporting of administrative and technical matters shall be via electronic mail (e.g., ARPANET). It is anticipated that much of the interim review and testing of code will make use of electronic mail capabilities.

Upon completion of each project, a technical report, in addition to Ada code, will be delivered to discuss the following issues:

a. Problems encountered with the Ada language in implementing the project

b. Problems encountered with the design methodology utilized

c. What code was re-used from another source (e.g., in-house, the Ada repository, etc.)

d. What code has high potential for re-use in other projects

e. Descriptions of all software including filenames, number of statements, lines and comments in each compilation unit

f. Experiences gained in the implementation environment (e.g., what tools were available and used in the environment, what problems were encountered with the environment during implementation, etc.)

g. Lessons learned during the scope of the project

It is anticipated that development efforts will be "instrumented" to provide data on productivity, the distribution of effort across the development life cycle, etc.

3.0 PROTOTYPE PROJECTS

3.1 Command Language

The command language must provide a communication interface between a computer system and all its users. It must enable every user to solve problems on the appropriate semantic level. In the WIS, there will invariably be users of all levels of expertise who

must communicate with the system and with each other. Their work will be facilitated by a common, uniform, consistent interface.

The preliminary prototypes formulated by the task force will provide multiple options for program control and allow the user to perform the following:

a. Manipulate the environment.

b. Control his session and data.

c. Manage processes and resources.

d. Package primitive commands.

The prototypes will allow for the development of uniform, consistent interfaces across all application areas in WIS. The use of the Common APSE (Ada Programming Support Environment) Interface Set (CAIS) as a virtual operating system model ensures the hardware independence of input/output (I/O) equipment. The functionalities of the command language capabilities will be partitioned into packages to facilitate maintenance.

3.1.1 Emulation of CAIS Form Terminal by a CAIS Page Terminal

The facilities available in a CAIS form terminal can be effectively emulated by a page terminal in host software. This package will provide a low-level forms capability upon which a more sophisticated forms package for data entry and command operations can be built. The Naval Ocean Systems Center (NOSC) tools, ANSI_VIRTUAL_TERMINAL and FORM_GENERATOR_SYSTEM, can serve as a basis for this task. A critical problem here is the determination of how to handle form terminal screen editing keys that only have a "local" meaning on a form terminal, i.e., they are never transmitted to the host.

3.1.2. Command String Parser/Interpreter

This prototype will be built on CAIS page terminal capabilities and could use the CAIS Node Management facilities to describe the command language. It would be greatly simplified by the availability of the Ada Command Language Interpreter as developed by GTE, Phoenix. Some of the capabilities provided by this prototype are as follows:

a. Facility to build, maintain, analyze, and document command languages

b. Uniform keyboard interface for a page terminal that supports editing the input line, spelling correction, name completion, help, and explanation

c. Uniform keyboard interface for a scroll terminal that supports limited line editing, spelling correction, name completion, help and explanation

d. Interpreter that invokes underlying functional capabilities (either system or application level)

e. Interpreter that invokes underlying functional capabilities (either system or application level)

f. Response library that maintains documentation about error responses (both messages and "signal conditions")

This task will provide a user-friendly development environment and will increase productivity.

3.1.3 Standard Command Language Implementation (Using the Command String Parser)

There are two command language standards that appear viable: the Conference on Data System Language (CODASYL) Common Operating System Command Language (COSCL) and the American National Standards Institute (ANSI) Operating System Command and Response Language (OSCRL). OSCRL is scheduled for completion in 1987 although committee members feel that work will be done much before then. COSCL is not a standard but rather a recommendation of CODASYL. It is published and is about to be enhanced with a menu- and form-based video versions. We need to determine which of these verions to recommend to WIS and then have an implementation built using the CAIS capabilities and the command string interpreter defined by task 2. This work is proposed by the WIS Foundation Technology Task Force group on Operating Systems.

The product will enhance a user's effectiveness in communicating with the system and with others about the system.

3.1.4 CAIS Form Terminal Version of Standard Command Language

Moving a command language from a page/scroll terminal to a form-based environment takes significant effort. The CODASYL COSCL committee is doing this for VIDEO COSCL. ANSI may follow suit for OSCRL. The NOSC tool, VIDEO, could be useful for this task. The size of this task will depend on the command language chosen and whether the responsible committee for the language has generated an appropriate VIDEO version of the language.

3.1.5 Definition of Form and Menu Tool for Bitmap, Page, and Form Terminal

This tool would be capable of generating a uniform editor-style user interface. All data in the system would be editable through forms. Users would access objects, edit them, or perform operations on them. This is the style of interface seen

in SMALLTALK*, STAR*, Macintosh*, and other advanced systems. Multiple windows and associated processes would be supported. Various graphic attributes, several types of graphical output (i.e., graphs, lines, video images, and text), scrolling, inter-window communication, and a variety of input devices would be controlled. A CAIS-like definition of a bitmap terminal would be generated. While different capabilities would be supported for various classes of terminals, the system would be capable of doing a reasonable job of interpreting a form for any terminal class. A key feature of this package would be that it would know about the data types and operators that are accessible to the user. Typical forms packages, like form terminal, deal only with alphabetic, numeric, and alphanumeric data and editing operations for these kinds of data.

3.1.6 Manual of Good Interface Style

Based on an analysis of the command language taxonomy and issues, the Manual of Good Interface Style will produce a set of guidelines to ensure that human factors engineering will be optimized in a WIS application software interfaces.

3.1.7 Tools for Uniform, Consistent User Interfaces

Using the manual described in paragraph 3.3.6 as a basis, the interfaces to new or off-the-shelf products will be designed in such a way that they will be WIS identifiable, and easy to learn and use.

3.1.8 Status

The preliminary specifications are undergoing analysis and reformulations. Two additional iterations are expected before final specifications are ready in the September 1985/December 1985 time frame.

3.2 Software Design, Description, and Analysis Tools

Careful attention to design is essential to the achievement of the important WIS goals of software reliability, efficiency, maintainability, and portability. Automated support can contribute to a substantial improvement in the design process. Many promising concepts for design automation remain unexploited. Prototypes and projects in this area will examine and demonstrate some of those concepts.

*SMALLTALK is a registered trademark of the Xerox Corporation.
*STAR is a registered trademark of the Xerox Corporation.
*Macintosh is a registered trademark of the Apple Computer, Inc.

Prototypes and projects are currently planned in the following areas:

a. Advanced Ada PDL

b. Ada PDL tools

c. Graphics support for design

d. Software metrics

3.2.1 Advanced Ada PDL

The objective of this project is to develop an advanced Ada PDL and basic supporting tool set for use in developing WIS software. This advanced PDL will be based on the current WIS Ada PDL Standard and will be extended by means of an annotation language such as ANNA. This will provide the basis for effective and rigorous testing, analysis and verification capabilities, and promote the reusability of PDL design elements.

The chief products of the project will be a PDL reference manual that succinctly and rigorously describes the syntax and semantics of the advanced PDL, an instruction manual describing its use, and a small number of tools including a PDL parser.

3.2.2 Ada PDL Tools

The objective of this project is to provide automated tools for the transformation, dynamic and static analysis, management, and documentation of designs expressed using an Ada PDL. Though primarily intended for use with the Ada PDL being developed as part of the "Advanced PDL" project, these tools will be sufficiently flexible to permit their use with other Ada PDL's.

The project will use the advanced WIS Ada PDL in the development of the tools. Where feasible, the techniques developed in the other projects will also be applied.

3.2.3 Graphics Support for Design

Software system design descriptions, especially descriptions of architectural designs, frequently have a graphical component. For example, control interactions in both sequential and parallel systems are commonly described by some variant of a flow chart. Other graphical descriptions that are of direct utility in software system design are structure diagrams, data flow diagrams, communication networks and hierarcial graphs. With the notable exception of the Tell software development support system, little support has been provided to software designers in the area graphics-based, man-machine interaction.

The intent of this project is to bring sophisticated graphics systems to bear on the problem of design description development,

modification and display. The product is to be a graphics-based software design workstation that provides to software designers the same level and nature of support that circuit designers enjoy through the availability of computer-aided design (CAD) systems.

3.2.4 Software Metrics

The objective of this project is to provide tools to allow for meaningful measurement of the quality of software design and code. Measurement can bring much needed visibility to the software product which is essential for effective project management and control. Metrics provide the opportunity to assess a software system's basic quality before investing in implementation. They can help to point out deficiencies within a design as well as to compare the quality of alternative designs. By providing specific, quantitative information, metrics can serve as valuable aids to making fundamental technical and managerial decisions.

This project involves the identification of metrics to assess the quality of a software design and implementation. This project also involves the development of tools to collect and analyze the various measures and to present the data in a form which is useful to the personnel associated with a software project. Finally, the project includes a demonstration of the resulting metrics capabilities on a pilot basis.

3.2.5 Status and Plans

We expect to complete detailed specifications and statements of work in each of the above areas by September 1985.

In addition, we are investigating a number of other areas and we expect to be able to define prototypes or projects in several of them. Those areas include reusability, documentation, test and analysis, verification, and object-managment systems.

3.3 Text Processing

The goals and objectives for the Text Processing Systems Task Force are to develop a complete Ada-based "document management" system with capabilities for, but not limited to, word processing, providing output with multiple type fonts, user-oriented "help" messages tailored to the operations being performed, and user expertise. In general, critical design issues include the following:

a. Provision for compatability with multiple subsystems that may or may not have been completely defined/developed

b. The ability to use a variety of I/O devices

c. The capability of using textual syntax/semantics to provide assistance in document preparation

d. The ability to provide user-oriented "help" based on analyses of operations being performed (for example, code being executed) and user history/expertise

Prototypes currently under consideration include the following:

a. Automatic generation of text-based systems

b. User interfaces for text processing

c. Integrated packages of writer's workbench tools

Emphasis will be in the direction of "what you see is what you get" (WYSIWYG) systems which permit multiple views with electronic documents and reasonable quality paper documents receiving the first specifications. All specifications will be extensible. These systems should assume and plan for advanced I/O technologies, should be I/O device independent, and should use virtual I/O devices.

3.3.1 Automatic Generation of Text Editors/Formatters

This project involves the design and implementation of prototype systems for automatically generating text editors and formatters across a variety of computers, input devices, and hard/soft copy output devices. One approach is to use a compiler writing model, such as grammars with action routines written in Ada or attribute grammars, for describing an editor/formatter, and to develop appropriate processors to generate systems from these descriptions. This syntax-directed structure editor should handle objects such as standard text, directories, mailboxes, core images (debuggers), and window managers. It should be able to take advantage of the graphics capabilities of the larger system of which the editor is a part. The system should have a common front end and set of editing functions, with specialized features for specific objects and applications. Its syntax-directed user interface should contain help facility, menus, grammar completion, and be extensible.

3.3.2 User Interfaces for Text Processing

The aim is to specify Ada packages for common document preparation tasks and I/O interfaces. Examples are basic string handlers, window managers, line and page break modules, command language interpreter interfaces, and database interfaces. The initial efforts will center on military joint message forms and Ada programs. These interfaces should be syntax-directed and contain help facilities, menus, and be able to perform grammar completion as well as be extensible.

3.3.3 Integrated Packages of Writer's Workbench Tools

The aim is to specify Ada packages for common tools necessary for a text processing system to provide to a writer. This writer's workbench includes tools for spelling error detection/correction, style analysis, indexing, bibliography database, on-line dictionaries, and glossaries.

3.3.4 Status and Plans

These issues are being looked at with the goal of a first draft of specifications to be available sometime in the August/September 1985 timeframe and version 1.0 prototypes available in late first quarter/early second quarter fiscal year (FY) 86. These functional specifications would be in a form similar to a short user manual. They would be in enough detail not only for someone to build a subsystem in Ada packages but also for the task force to evaluate the results. Already, the need for two grammars, with associated action routines and/or attributes, has been identified. One grammar would be for objects and the other for the command language. The specification method and means for connecting these two grammars are still the subject of research.

3.4 Database Tools

The goals and objectives for the Database Task Force are the definition of a standard, portable Ada-commercial off-the-shelf (COTS) database management system (DBMS) interface and development of an Ada-DBMS. The Ada-DBMS provides for the following capabilities:

a. Database definition, including the provision of multiple views (i.e., relational, network, hierarchic) of the same database

b. Efficient retrieval and update of individual objects or groups of logically related database objects

c. Authorization control to the field or attribute level, capable of expansion to provide multi-level security control (e.g., prohibiting a query that is authorized to access aggregated data at a given classification from generating secondary accesses to data objects at a higher security classification)

d. Multiple interfaces, including programming and query languages, screen-oriented command language/displays, bulk load/unload facilities, report writer and application generator

e. Multiple user access with backup and recovery

f. Database administration and control

Critical design issues include the following:

a. Support of fully distributed access, including partial or full replication of objects

b. The ability to incorporate "database machines" or "back end database processor" technology for retrieval operations

c. Expert system interfaces

d. Verifiable security protection (ability to operate in a multi-level security environment)

e. Efficient processing of multiple views/schema

f. Hardware independence and portability to include optimization of retrieval strategies

Thirteen potential database management system tools/components to aid in the creation and tailoring of support for a range of high-level systems were initially considered for prototyping. These included the following:

a. An indexed file access package

b. A concurrency control package

c. A workstation relational database system package

d. A multi-user relational database system package

e. A relational database design tool kit

f. A view definition facility

g. A view query and update facility

h. An authorization package

i. A database query optimizer/compiler

j. A distributed database access package

k. Distributed database protocol management

l. Database operational tools

m. Database application tools

Based on a set of criteria which emphasized independence, general understanding of functionality and implementation method, and immediate usefulness, three components have been identified as candidates for early implementation: Indexed File Access Package,

Workstation Relational Database System, and Database Application Tools. It appears desirable to develop these components in parallel.

3.4.1 Indexed File Access Package

This project involves the design and implementation of an indexed file access package in Ada to provide a flexible and standardized data storage interface for database components as well as programs not requiring the power of a database system. The motivation for early prototyping of this component is to provide a de facto standard for structured files. Without a standard, a great variety of mappings from data processing and database records to external storage will be developed and used within specific projects. An implementation is needed to support any parallel paper standardization guideline.

An indexed file access package should provide access to records of variable length. The records can include fields of variable length. The records should be accessed by one or more symbolic keys, by either random or sequential access. This requires multiple indexes which should provide a symmetric interface, although one index may be chosen as a clustered index to provide more efficient sequential access through that index without any effect on functionality. Any index may be specified as unique to require that all records have unique values. Optional retrieval by record identifier and retrieval of lists of record identifiers, corresponding to records with a particular index value, is needed to facilitate efficient implementation of boolean queries referencing multiple fields. When used in a controlled manner, the record identifier can be used to improve the efficiency of inter-record references in databases. The architecture should permit a high degree of concurrent access in a manner which reduces non-inherent deadlocks. This package would be useful not only for databases but also for ordinary application programs.

3.4.2 Workstation Relation Database System Package

The aim is the specification of a workstation relational database system package in Ada that implements a fully relational, single user database system. Workstations are a vital component of modern software development. The simple, single user environment simplifies many implementation issues. The generous amounts of memory available on modern workstations permits adequate performance to be attained with simple algorithms.

A relational database system package would provide a direct usable application within this framework. A version for a workstation would depend on the file component to provide a fully relational single site database system. It should provide transaction management facilities, including rollback on abort, logging, and recovery mechanisms.

3.4.3 Database Application Tools

The aim is the specification of database application tools, such as report writers and screen-oriented interactive query, update facilities to provide user-friendly interfaces, and permit rapid exploitation of database technology by non-programmers. Experience on current commercial microcomputer systems has demonstrated that databases, with spreadsheets and word processing, have provided direct non-programming access to computing. A powerful and flexible report writer package would substitute report specification for much application programming. An interactive ad hoc query/update interface would provide sophisticated users with powerful access to the database. An interactive application generator would facilitate rapid prototyping of screen oriented query and update applications.

Object-oriented Ada design permits considerable flexibility in implementation and use of the system. For example, natural language query interfaces and expert systems can be built on top of this system. Components of the system may be placed on their dedicated machines to take advantage of database machine (DBM) technology.

3.4.4 Ada-DBMS Interface (Ada/Structured Query Language (SQL))

The objective of this effort is the development of a standard for utilizing COTS DBMS's from Ada programs. The availability of such a standard would simplify and streamline system development efforts as well as improve the transportability and maintainability of the systems thereby developed. The Ada Data Definition Standard comprises two standards, the Ada Database Standard and the Ada Data Manipulation Standard. The Ada Data Definition Standard is the first part of the Ada Database Standard. Given a requirement to process data, the Ada Data Definition Standard will provide guidelines and direction for using Ada data types and structures to represent that data. The primary motivation will be to structure data using Ada records, with components that may be arrays, arrays of records, etc.

The Ada Data Manipulation Standard is the second part of the Ada Database Standard effort. Given data structured as by the Ada Data Definition Standard, there are certain operations that will be required upon that data. This effort will define those operations and will specify their desired point of implementation within a system (e.g., universal package, application domain package, individual application program, etc.).

3.4.5 Status and Plans

These issues are being investigated with the goal of a first draft of the specifications to be available in the August/September 1985 timeframe and version 1.0 prototype of the Indexed File Access Package available in late first quarter or early second quarter FY 86.

An Ada programming language access to the draft proposed American National Standard (dpANS) Database Language SQL has been defined. Called Ada/SQL, it is compilable, native Ada. In addition, the format of the Data Definition Language (DDL) has been defined and partially implemented. This, too, is compilable, native Ada, which can be "with'ed" by the users' Ada application program. This method of defining a DDL gives the advantage of additional automated type-checking, performed by the compiler. A set of programs has been defined that takes as input the DDL, and produces as output all of the underlying packages needed to implement this interface for a particular instance of the DDL. These programs are completely portable, and render negligible the time needed to define the interface in terms of the DDL.

A test data generation program is being developed in parallel. This program will take as input the DDL defined by the user and create meaningful test data for the database. The test data so generated may be as large as several gigabytes. Test and evaluation of the Ada/SQL implementation is being conducted. The Unit Status and Reporting (UNITREP) database build module is being redesigned and rewritten to replace the previous IDM-500 Database Machine interface code with Ada/SQL.

3.5 Operating System

The set of Ada prototypes being specified by the WIS Operating System Task Force will provide a highly reliable and portable operating system. This system will adhere to the CAIS standard "node" model, include discretionary and mandatory access control at the B3 level, and consider uni-processors, multi-processors, and multi-computer systems. While the WIS environment is a collection of local area networks (LAN's), connected by one or more wide area networks (WAN's), prototypes will concentrate on a LAN. However, the design and implementation will not preclude, nor make difficult, interactions with other LAN's of WIS. WIS is a demanding environment that must have the following attributes:

a. Fault tolerance

b. Survivability

c. Multi-level security at the B3 level

d. Portability of applications and system software across a wide variety of machine sizes and types

e. Single- and multi-thread machines

f. Multiple priorities

g. Realtime processing

h. Fully integrated databases

Twelve prototype projects are currently being specified. These include the following:

a. Kernel

b. Program Execution Support Module

c. File Manager

d. Authentication Server

e. Time Synchronization Agent

f. Transaction Manager

g. Inter-Process Communication (IPC) Support

h. Alias Processes for Access to Remote LAN's

i. Print Server

j. I/O Drivers

k. Multi-Window System

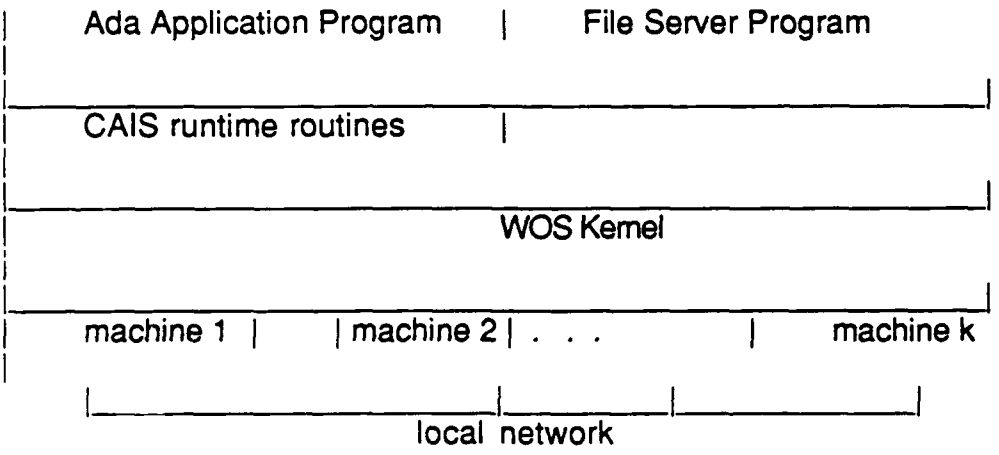
l. Logging and Auditing

Brief descriptions of the first ten prototypes are presented in this summary.

3.5.1 WIS Operating System (WOS) Kernel

The WOS kernel is a distributed kernel that implements an abstraction of memory, processing and communication suitable for implementing CAIS-like process nodes as well as external file and structural nodes. The kernel is designed to provide minimal facilities required to meet security, fault tolerance, performance and CAIS compatibility requirements. Facilities that need not be implemented in the kernel are implemented in server nodes that execute outside the kernel as well as by runtime procedures that execute in the address space of the invoker. For instance, the CAIS function STANDARD_ERROR may be just a simple routine linked directly with the invoker, returning a value stored in the address space of the invoker. Other functions, such as SPAWN_PROCESS, use kernel facilities to create a process node (address space) and allocate the needed resources with the required protections and security control.

The following illustrates the layering of CAIS routines on the kernel implementation with a service module.



The kernel logically extends across all the machines in a WOS cluster, although in reality, a copy of the kernel (or some version) executes on each machine in the cluster. The basic kernel interface is augmented by CAIS runtime routines to provide CAIS process node, structural node, and file node operations. In the case of file nodes, there are one or more file server programs that actually implement files. The file server programs may execute on the same or different machines (or both) with transparent access to local and remote file nodes provided by the kernel facilities plus the CAIS runtime routines. In general, the kernel is viewed as an protected runtime library that implements routines and data structures that cannot, for security, fault tolerance or performance reasons, be implemented in the CAIS runtime packages. These packages execute addressable to applications.

3.5.2 Program Execution Support Module

This prototype will provide a set of Ada packages that supports program execution for the WIS environment. The main interfaces of interest for the Program Execution module are the compiler, linker and loader outputs, the command language (CL), the kernel, and the storage management packages. It is assumed that the CL is sophisticated and that it can support invoking typical functions. Typical functions include compiling, linking, loading, and executing programs, as well as creating, deleting and manipulating files and libraries, etc. The kernel is assumed to support the compiling, linking and loading aspects of program execution. Note that the complier, linker and loader outputs together with the kernel support defines the contents of the virtual address space, e.g., where the data is. Data includes code, external symbols, global variables, heap, stack, etc. It also resolves questions of what parts, if any, of the operating system are in the virtual space and where. The kernel and storage management modules support the idea that all resources are "objects" so that the Program Execution module can issue "GET

OBJECT" calls for acquiring resources such as main memory and files.

The Program Execution Support Module will address the following issues:

a. Distributed resource management

b. Distributed scheduling

c. Deadlock avoidance, detection, and resolution

d. Support of the required attributes of a WIS environment (in the context of program execution)

3.5.3 File Management

The file management server will provide an efficient, reliable file system supporting file locking, page-level locking, replication and recovery and atomic update. Each file is a sequence of fixed-size "blocks" or pages whose size is known. For efficiency reasons, the blocks should be one kilobyte or larger with four kilobytes looking very attractive. The blocks associated with a file represent the data most recently committed to that file.

3.5.3.1 File Access

To access a file, a file is "opened." The open file represents a "version" of the file blocks. Two options are available. The open file may represent the data blocks as they were (those committed) at the time the file was opened except for any modifications made using this open file. In particular, changes (even if committed) made via other open files are not apparent in this open file. The alternative mode is one in which changes to file pages appear in the open file once committed by others in addition to any changes made to this open file. These two modes are perceptually equivalent under appropriate locking assumptions. An open file is the unit of atomic update/abort provided by the file system, as part of the close operation. File closing also releases all locks. The file system also implements a Save Point operation that effectively provides a commit without losing access to the file, as with the close operation.

3.5.3.2 File Locking

Lock management provides locking of files as well as blocks within files. The lock management for a file resides in the file manager implementing the file. Consequently, there is not a "floating" global lock manager. Co-locating locks with the file managers also allows us to combine lock operations with read operations, thereby reducing communication costs when operations are remote.

Lock modes include shared-read and exclusive-write. A lock is associated with a particular transaction identifier and user account. For example, a write to a locked block fails unless the writing task specifies the same transaction identifier as recorded for the lock plus has the same user account as the creator of the lock. The transaction association (versus a process/task association) allows multiple tasks involved in a transaction. The account association is for protection between users.

It is necessary for file managers to have a reasonable degree of autonomy. A file manager is committing resources to a transaction in the form of open files and their associated open files. For this autonomy, it is necessary for a file manager to reclaim these resources under certain situations. This means that the file manager must be able to effectively abort the open file, releasing locks and undoing any changes resulting from this open file. This undo facility is clearly required or else file aborts of this nature would leave the file in an inconsistent state.

3.5.3.3 Backup and Recovery

The file managers must, for autonomy again, provide for backup and recovery independent of application programs (e.g., a DBMS). The file managers must perform journaling to support atomic transactions on files. Shadow paging for atomic update is unacceptable in database files because of the negative influence on disk contiguity. Therefore, old pages must be logged when updated and restored from the log if the file is aborted.

3.5.3.4 File Replication

A file may be defined to be replicated. The file replication is for survivability and performance. Therefore, this is specifying physically dispersed replication as opposed to disk mirroring or closely coupled replication on the same machine.

An application program must be able to specify/suggest the sites at which the files should be replicated so as to co-locate related data, e.g., relations that commonly are joined. There is some issue as to what the file system does when the suggested replication locations cannot be used. It may be simplest to return an error so the application program can pick an alternative or specify "don't care".

When a replicated file is accessed for reading, the operating system provides access to the replica with the least cost to access. The operating system may take into account communication speeds, processor load and other factors.

3.5.4 Authentication Server

The authentication server provides authentication, key distribution, and some aspect of naming. We first describe the security model in brief.

3.5.4.1 Security Model

WIS must implement a multi-level secure system corresponding to at leave the B (mandatory protection) level, ideally B3 with feasibility of going to A1 with the appropriate verification tools.

The system has a designated trusted computing base (TCB). One avenue in attempting to make the system verifably secure is to make the TCB as small, well-defined and well-structured as possible. Clearly, the kernel is part of the TCB. It provides normal processes that execute at different security levels and ensures that there is not interaction between processes at different security levels. Actually, the kernel operations may also allow for "read up," higher clearance levels reading from lower levels. However, we do not provide write-up so as to simplify the integrity problems with the system. The kernel also supports reclassifier processes that may move data between security levels. In particular, reclassifiers are used to implement controlled "write-down." Each reclassifier process is also part of the TCB since misbehavior of a reclassifier consititutes a security violation.

Thus, the kernel enforces multi-level security for entities with known security levels. It provides also the creating of new processes with security levels inherited from their creators. However, it is the authentication server that assigns a security level to a new entity. Clearly, the authentication server is also part of the TCB. The authentication server performs several services, including authentication, key distribution and security logging.

3.5.4.2 Authentication

The authentication server maintains a collection of accounts which are created by the security officer at a particular clearance level. A user must authenticate himself with the authentication server before making any non-trivial use of WIS. This entails communicating account name and password (or key) to the authentication server. The communication itself might be encrypted using the password as a key. The authentication server then decrypts the message and checks the correctness of the password. One rather inefficient but secure approach is for it to attempt to decrypt it with every account password and check correctness in this fashion. This would avoid sending the account name in the clear.

Assuming the authentication request was valid, the authentication server communicates with the kernel executing the requesting process and requests that the kernel change this process to the specified account and security level. This relies on secure, unforgeable communication between the authentication server and each copy of the kernel. In particular, there must be absolute safeguards against an impostor authentication server. The authentication server logs all such authentication requests, whether successful or not.

3.5.4.3 Key Distribution and Encrypted Communication

The authentication maintains a secret secure key associated with each account. This is used primarily for secure communication with the authentication server. To communicate with other service modules securely, the authenticaton server provides "conversation keys." This minimizes the use of the principle keys. Compromise of a conversation key is less critical than a compromise of the principle account key.

3.5.4.4 Security Logging

The authentication server should provide a logging facility that records all security-sensitive events. These actions include those taken by itself such as each authentication request and its result. It also includes actions the kernel takes, such as creation of reclassifier processes, messages from the authentications server and any attempts at security breaches.

3.5.5 Time Synchronization Agent

The notion of which event "happened before" another one is usually based on physical time. In a distributed system this would implicitly assume that some universal time can be defined and observed identically from various locations. This is impossible. Practically, one may approximate some universal time with a given accuracy, and use this approximated universal time to develop a relative ordering of events, either a partial ordering or a total ordering - whatever is required.

Consider that a logical clock can be described as a function F which assigns a number to any action initiated locally. Such logical clocks can be implemented by a simple counter. Now consider a distributed system where each producer process owns a logical clock. The problem then is to guarantee that the system of clocks satisfies a condition Z so that a particular ordering may be built on the set of actions initiated by the producers. In general the ordering is not unique and may not be equivalent to a chronological ordering. The objective of the Time Synchronization Agent is to obtain a unique physical time frame within the system so that consistent schedules may be derived from a total chronological ordering of actions occuring in the system.

When multiple physical clocks are involved (as is true in a distributed system), it is not enough that the clocks run at approximately the same rate. They must be kept synchronized. It is possible to synchronize the local clocks of various processors in a distributed environment. However, an accuracy is limited roughly by the sum of errors which accumulate because of different clock rates in each computer, and errors arising from the uncertainty about the time for communication between machines. The clocks must be synchronized so that the relative drifting of any two clocks is kept smaller that a predictable amount.

3.5.6 Inter-Process Communication

A @@i[pipe] is a synchronized file that allows one or more readers and writers to transfer data, using the pipe as a bounded buffer. A familiar implementation of pipes is the UNIX* operating system.

The pipe facility provides symmetric inter-task communication in the I/O model of communication. The program or task that writes its output to a file can be connected by means of a pipe to the input of another program or task that reads from a file. A series of two or more tasks interconnected by pipes to run concurrently is commonly called a @@i[pipeline]. Such a task pipeline executes in parallel in a similar manner to the pipeline structure used in a hardware processor. Each "stage" of the pipeline processes its input data and passing the resulting output to the next stage in parallel with the execution of the other pipeline stages.

The pipe facility complements the asymmetric inter-task communication provided by the Ada rendezvous mechanism. With the rendezvous mechanism, one task is a client who invokes entries; the other task must be a server who passively waits for an accept or select statement to complete. In plumbing terminology, the client has a male sex connection while the server has a female sex connector. In this analogy, a pipe provides a connector with two female ends, allowing two tasks with male ends (clients doing entries) to be plugged together. In addition, a pipe provides some amount of buffering to allow greater concurrency between the two communication tasks.

While there are other options for the design of such a "sex matching" inter-task facility, we choose to implement this facility in the file model, using the CAIS-specified "queue files" as the program interface for applications. A queue file in CAIS represents a sequence of information that is accessed in a

*UNIX is a registered trademark of the Bell Laboratories.

first-in, first-out manner. There are three kinds of CAIS queue files: solo, copy and mimic. The solo queue file corresponds to the UNIX pipe. It is empty when created initially and then operates like a standard queue. All writes append to the end of the file and all reads are destructive reads of the beginning of the file. A copy queue file operates like a solo queue file except that its initial contents are copied from another specified file. A mimic queue file is similar to the copy queue file except that writes to the mimic queue file are also appended to the file from which the queue file received its contents. Refer to the CAIS Specification Manual for further details of the creation of queue files and operations on queue files.

We propose that queued files be implemented by a concurrent Ada program that runs in one or more dedicated process nodes within each WIS cluster. In fact, there are performance advantages to having a queued file server or pipe server on every WIS node that uses queued files. In particular, it is more efficient for two tasks or programs running on the same machine to communicate by a queued file implemented on that same machine, rather than transmitting the data across the network. Furthermore, it is advantageous to have the queued file implementation on the same machine as at least one of the clients of the queued file, in the case where the clients (the reader and the writer) are running on different machines. This reduces the communication from two network transfers to one. However, remote clients can access a queued file implementation so having a local implementation of queued files can be viewed strictly as an optimization.

A simplified implementation of queued files or pipes is possible if we assume that the reader and the writer of the queued file is fixed at the time the queued file is created. Under this assumption, either the reader or the writer can include in its runtime support (within its address space), code for "reversing sex" to match the other clients plus some buffering. We reject this implementation, despite its performance advantages, because it does not allow one to change the reader or writer and significantly complicates the runtime code in each client. Note that this does not preclude simple "bounded buffer" facilities for tasks sharing data in the queued file model within one address space.

We propose that all three types of queued file be implemented by an Ada concurrent program structured much like the file server. It could be included in the file server program although it is appealing to have this facility available without having the entire file system code resident. In the simplest case, the queued file server consists of a simple task that provides entries for creating, deleting, opening, reading, writing, closing and querying queued files. The data in each queued file is simply stored in virtual memory. For greater potential parallelism, there could be one task per queue file, allowing the use of the

select mechanism to synchronize readers and writers, similar to that shown in paragraph 9.12 of the Language Reference Manual (LRM).

Copy queued files simply required an initialization of contents from a given file and are otherwise implemented the same as solo queued files. Mimic queued files require that writes to the queued files also result in writes to the original file, another small extension on solo queued files.

This Ada program is expected to be fairly modest in size and might most reasonably be combined with the file system/storage management programming effort.

3.5.7 Alias Processes

Each cluster has one or more gateway machines that connect the cluster to WAN's and, therefore, to other clusters. Unlike conventional datagram gateways that simply provide packet routing (and little else), cluster gateways are required to take an active part in insulating the cluster from outside considerations. These considerations include differences in communication protocols and characteristics, security and reliability assumptions. The gateway really does have to function as a "gate" that can close selectively, not just a mindless pipe to the outside.

For efficiency, within a cluster, network communication is optimized for local network characteristics or logical local networks. A logical local network is one or more physical local networks connected by badges such that the existence of multiple physical network is more or less transparent to the hosts connected to the network. For communication outside the cluster, the gateway implements a local alias task that represents the remote task with which communication is to take place. Similarly, communication coming into a local network is handled as originating from an alias task in the gateway. This basic model as several benefits. First, communication with tasks outside of the cluster appears the same as communication with a local task because of the local alias task. Additionally, there is no need to compromise local communication with a local task because of the local alias task. Thus, there is no need to compromise local communication to make wide-area communication possible. Consequently, local cluster communication is very efficient.

Second, inter-cluster communication cannot occur without the gateways agreeing to create the requisite alias tasks. Because the creation of alias tasks can be handled at a fairly high level, this provides a reasonable security control over inter-cluster communication.

Finally, because the gateway is explicitly involved in translating between local communication and wide-area communication, it can serve to isolate the local network from

outsider failures and errors as well as contain local cluster failures

A cluster gateway is structured as a half-gateway in the sense that it translates from the local network protocol to an inter-network protocol (in the truest sense of the term). That is, the inter-network protocol is primarily used between networks as opposed to covering a collection of networks. On the local network side, the gateway appears as a server that provides access to extra-cluster facilities. On the inter-network side, the gateway appears similarly with the addition of explicit data transfer primitives and a more general data packets interface.

The gateway server registers as a service in a cluster the same as other services, running on top of a standard kernel. The kernel must allow the gateway server to receive packets addressed to alias tasks as well as sent to any groups to which alias tasks belong. In addition, the kernel must allow the gateway server to send out inter-kernel packets to local tasks. Basically, the gateway server implements the inter-kernel protocol for local alias tasks, receiving packets addressed to them, taking the appropriate action, and then responding

When a Send packet is received for an alias process, the gateway server checks whether this is a retransmission. If not, the gateway server transmits the packet on to the remote cluster and task associated with this local alias. The remote gateway, on receiving the Send packet, translates it to the appropriate local task, translates the sender task-id into a local-to-this-gateway task-id and forwards the packet on to the local task as though it was a local Send. Retransmissions are filtered out if the connection between cluster gateways is reliable. In any case, the rate of retransmission across wide-area links can be modified by the gateways to match the performance characteristics, independent of timeout values and retransmissions on the local network cluster.

The cluster gateways could make good use of an internet multi-cast facility, as is being currently proposed.

The contractor should develop a specific design based on this general outline with attention to security, fault tolerance and general survivability. Ideally, the inter-cluster directory should make use of distributed integrated database technology developed as part of the rest of WIS.

3.5.8 Transaction Manager

The transaction manager service interacts with the kernel and file manager to provide fault tolerance. It requires a distributed file server that supports file locking and replication, and will include facilities to support fault tolerance, concurrency control, recovery and checkpointing.

A transaction is an atomic unit of execution. It is a sequence of operations which are either completely executed (commit) or not executed at all (abort). It has the properties of atomicity, durability, seralizability, and isolation. The notion of transaction is therefore related to the problems of fault tolerance and concurrency control. The goals of transaction management are the efficient, reliable and concurrent execution of transactions. These goals are strongly interrelated; therefore, some tradeoffs are needed to design and implement the system.

The base technique for implementing transaction in the presence of failures is based on the use of logs. A log contains information for undoing and redoing all operations which are executed by the transactions. These logs should be kept in stable storage. The write ahead protocol and the careful replacement strategy should be used to record a log and to update stable storage, in that sequence.

To improve the degree of recoverability and to increase the degree of concurrency, the concept of nested transaction has been proposed as a generalization of the basic transaction. Several nested-transaction models have been proposed in the literature. The ideas of nested transactions has also migrated into the design of Network Operation System (NOS) and distributed programming languages. This concept will be prototyped for use in the WIS storage management server.

3.5.9 Printer Server

The printer server is basically just an output service. Data is written to the printer server, just like a file. However, the printer server also has a separate class of operations to query its state and modify its operation

In this vein, there are three aspects to the printer server: the data output, print job queuing query and modification, and printing parameter control and query.

The printer output is handled using standard Ada file writing with access to the printer file established using the same interface as specified for ordinary disk files, and hopefully similar to that in CAIS. The printer file is actually a virtual printer which is then spooled for the printer.

To output to a printer, the program opens a file using a printer server name. This open file is just a ordinary disk file created by the printer server. When the file is closed, the file is queued to be printed, and presumably deleted once printed.

A particular printer server may have several printers plus print jobs arriving at times faster than they can be printed. Therefore, as a standard, the print jobs are queued until a printer is available. A large printer server would have its own file system, presumably using standard file system software.

Operations are required for querying the print queue, changing the priority of print jobs, deleting a job from a print queue, aborting a print job and restarting or backing up a print job (to handle the case of the printer jamming, for instance).

A set of operations need to be provided to control how jobs are actually printed. These include operations for changing printer characteristics, changing fonts and paper, and other aspects of the printing

3.5.10 I/O Drivers

There is a requirement for a common interface to all I/O devices, with the device "name" or "address" as part of the call. The use of parameters and their sequences shall be as uniform as possible among the various devices and drivers. The device name will be mapped to a real device in the common interface module. A device "address" may be symbolic and may be similarly mapped. In general, this mapping will be transparent, and the user need not know much, if anything, about the device to which the mapping is made. Furthermore, the mapping may be used to redirect the user's I/O transparently.

Each driver should, to the maximum extent possible, depend only on the device and not on the host system.

Drivers shall be written for all devices to be used anywhere in WIS, and to work with each machine used in WIS which connects to any of these devices. To the greatest extent possible, code for the drivers shall be common between devices; different code sequences should be written only when required. Drivers should be written initially for those devices currently in use or projected to be in use in WIS.

Following is a list of devices likely to be used somewhere in WIS

a. IBM 3330, 3340, 3340, 3350, 3380 disks, all common varieties of 8", 5" and 3.5" floppy disks, and commonly available small hard disks

b. Tape drives at 1600 bpi and 6250 bpi for all widely used computer systems (IBM, DEC*, Honeywell, Sperry, etc.)

c. New IBM cassette drives, card punch, card reader, and communications lines at all the usual speeds (e.g., 300, 1200, 2400, 4800, 9800, 19200, etc.)

c. Ethernet interface, standard ASCII terminals, bitmapped high resolution terminals, and graphics displays.

*DEC is a registered trademark of the Digital Equipment Corporation.

The I/O system shall provide a mechanism whereby one device may be substituted for by another, provided that the functionality is equivalent in a manner invisible to higher levels (except perhaps for performance). Such substitution shall even be possible across a LAN or a WAN.

To the maximum extent possible, interfaces for similar devices should be similar and standardized. For example, a "standard terminal interface" should be the front end to all terminals. All text-type terminals should support the same minimum set of commands, with more advanced terminals supporting additional features, defined as optional in the interface. Likewise, the disk interface should at the highest level provide a standard set of commands. At a still higher level, it should be possible to treat most or all devices as a serial bit or byte stream. Justification should be provided in each case as to why interfaces between devices of the same "type" or of different types are not the same. And where there are layers, with commonality above a certain point and differences below it, the location of that point should be justified.

3.6 Planning and Optimization Tools

In response to the Joint Chiefs of Staff Statement of Operational Requirements for the Joint Operations Planning and Employment System (JOPES), it has been determined that a decision support system is required. This system is named the WIS Smart Advisor for Planning and Execution of Decisions (WISSAPED). The WISSAPED will consist of three components: a Model-Based Management System (MBMS), a Dialog Generation amd Management System (DGMS), and a Database Management System (DBMS). The initial effort has concentrated on the definition of the MBMS and description of a hierarchy of modules which can be threaded together to form models.

There are a large variety of problem formulations that may arise as a result of the planning process. Examination of the problem formulations resulted in placing the necessary models into six classes

a. Linear systems

b. Non-linear systems

c. Networks and graphs

d. List processing

e. Time series

f. Stochastic

Each of the model classes were further subdivided to identify the hierarchy of tools required

a. Fundamental data structures

b. Primitive operations

c. Analysis tools

d. Solvers

e. Simulators

f. Optimizers

To a significant extent, each succeeding level uses the tools in preceding level

The initial efforts of the Planning and Optimization Task Force have been concentrated on the following efforts:

a. Formalize the structure of the WISSAPED.

b. Define and describe the hierarchy of tools.

c. Elaborate on one of the six model classes, e.g., linear system models

d. Develop specifications for tools which are necessary for producing "optimized" solutions for four of the more common classes of planning problems.

3.6.1 Generic Linear Programming Package

This prototype project is for the development of a generic Ada software package to solve problems of the Linear Programming class. The problem formulation is to:

minimize $f(X)=CX$
subject to $AX=B$ and $X > 0$,

where C is a row vector and X
and B are column vectors
$C=(c1, c2,\ldots,cn)$,
T

$X=(x1, x2,\ldots,xn)$,
T

$B=(b1, b2,\ldots,bm)$,
T

A is an mxn matrix

$$A = \begin{vmatrix} a11\ a12 \ldots\ldots\ldots a1n \\ a21\ a22 \ldots\ldots\ldots a2n \\ \ldots\ldots\ldots\ldots\ldots \\ \ldots\ldots\ldots\ldots\ldots \\ am1\ am2 \ldots\ldots amn, \end{vmatrix}$$

ci, bj and akl are real-valued,
and X > 0 means all xi are non-negative real-valued.

The generic package P will consist of several specialized subpackages Pi, each of which is tailored for specified ranges of n and m and joint characteristics of C,B, and A (A might have a tree structure, etc.). An upper limit for m of at least millions and for n of at least hundreds of thousands is required for sparse A. An upper limit for n of at least 400 is required for at least some applicable types of dense A. High quality (though not necessarily of production-quality) Pi should be included for A,B,C relationships that commonly occur in military contexts, especially logistics. "Quality" refers to maintaining accuracy while operating at high speed and efficiency in modern machines. It also refers to the reliability and precision of all Pi outputs.

Outputs: Output will include error estimates; sensitivity indices; and, where appropriate, diagnostic information pointing out especially troublesome spots in the Pi run and/or in the problem formulation for the Pi that ran it. Warning of computational disasters due to ill-conditioned problems will always be provided for the user as soon as possible.

Inputs: Inputs include n, m, A, B, C; accuracy bounds (if known) on the sets of bi, cj, and akl parameters; a menu-guided characterization of A and the A,B,C relations sufficient for automatic selection of a good Pi to run the problem; and output and stopping conditions on the run.

Methods: The method employed by each Pi (e.g., simplex, revised simplex, dual-simplex, Karmarkar, etc.) will be indicated. Its characteristic capabilities will be available before run time in both naive user and professional analyst terminologies. A succinct description of the method will also be available as a brief complement to the method's full documentation.

The generic P will easily be able to accommodate any additional Pi that might be added later.

3.6.2 Generic Mixed Integer Linear Programming Package

The purpose of this prototype project is to solve the mixed integer linear programming (MIP) problem which is formulated as follows

Minimize (cx + dy)
x > 0
y > 0

subject to Cx + Dy < b,
x integer,

where C and D are matrices and c, d, b, x and y are vectors of appropriate dimensions. By taking d and D to be null, this problem specializes to the "pure integer" linear program. The MIP package will be extensible to include a multi-attribute objective function by replacing the scalar cx + dy in (MIP) by Ex + Gy, where E and G are matrices of appropriate dimension.

Scope: The prototype tool will consist of several specialized subtools, each of which is tailored for specified ranges on the numbers of variables and constraints as well as specified relational characteristics among the MIP parameters c, d, C, D, and b. An upper limit of at least 800,00 variables and 200,000 constraints will be required

3.6.3 Generic Non-Linear Programming Package

This prototype project will develop a generic software package in Ada to solve problems of the form:

minimize F(x),

x a member of R"

$$\text{subject to} \qquad l \leq \begin{vmatrix} x \\ Ax \\ c(x) \end{vmatrix} < u,$$

where F(x) (the objective function) is a smooth non-inear function, A is a constant matrix, and c(x) is a vector of smooth non-linear constraint functions. An equality constraint corresponds to setting li=ui. The special case when F is linear and c is not present is the usual linear programming problem. Note that simple bounds and linear constraints are presented separately from nonlinear constraints.

Scope: The prototype package will be capable of handling three major size-classes of problems: small-scale problems having about five or fewer unknowns and constraints; intermediate-scale problems having from about five to a hundred variables; and large-scale problems having more than a hundred and perhaps even thousands of variables and contraints. This classification is not entirely rigid, but it reflects at least roughly not only size but the basic differences in approach that accompany different-size problems.

Outputs: Outputs include the solution and sensitivity information as requested; run analysis results such as matrix factorization statistics, iteration logs, exit conditions at various program stages, error estimates and the effects of various errors on solution precision; and a brief run summary.

Inputs: Inputs include the necessary problem parameters.

3.6.4 Generic Sequencing, Scheduling, and Assignment (SSA) Package

This prototype project will develop a generic software package in Ada to find optimum or quasi-optimum sequences, schedules, or quadratic assignments for standard deterministic and stochastic combinational problems

The standard deterministic sequencing and scheduling (DSS) problems include the traveling salesman and other routing problems; one-dimensional, bin-packaging problems, open-shop, flow-shop, and job-shop scheduling problems; and various critical path and transportation problems. Stochastic sequencing and scheduling is, for the most part, still too immature for inclusion in a prototype package.

The prototype package will include quadratic assignment problems (QAP) of the general Koopmans-Beckman type. The QAP are combinational optimization problems of the following form:

given a finite set N=§1, 2, ..., n† and two (n X n) matrices

$$A = (a_{ij}) \text{ and } B = (b_{kl})$$

with real entries. The matrix A is called a distance matrix; B is called a connection matrix. The problem is to find a permutation of the set N such that the sum

$$\sum_{i \in N} \sum_{j \in N} a_{ij} b_{f(i)f(j)}$$

becomes minimal

Subprograms for handling knapsack problems (KP) of any dimension are candidates for inclusion in the prototype package. A general KP is to maximize the value:

$$\text{Maximize} \quad \sum_{i=1}^{n} c_i x_i$$

$$\text{subject to} \quad \sum_{i=1}^{n} a_i x_i \leq b,$$

$$0 \leq x_i \leq u_i \quad \text{an integer.}$$

Scope: Most prospective military applications of an SSA package would involve large numbers of variables. Some packing algorithms have handled up to 10,000 items, some routing programs up to 500 cities, some DSS algorithms execute in times proportional to a low degree polynomial in the number of problem variables, etc. But small changes in a problem's formulation can lead to huge changes in its complexity. Thus while we can specify that our prototype's programs should handle the largest size problems that are feasible for modern computers and that these programs have at least some military applications, little more than this can be said at present.

The prototype should have a rich set of interactive capabilities. This will allow the use of multiple criteria and the selection of robust schedules that tolerate differences between models and their real-world counterparts sufficiently well as to have practical value.

The prototype should aim to knit together a significant number of DSS, QAP, and KP subprograms in an extendable and easily maintainable way. These programs might not involve annealing methods, and should not be organized as standard integer programs These standard integer programs are to be established in a separate prototype tool

3.6.5 Status and Plans

We expect to complete detailed specifications and statements of work in each of the above areas by September 1985.

As additional efforts, we will prepare detailed specifications and statements of work to develop Ada software for the models which will reside in the MBMS.

3.7 Graphics

A high-level conceptual model of a graphics support system for WIS applications has been developed by the graphics task force. The model is a pipeline showing the control of information and control between one or more application programs and the graphics display and input hardware. The model illustrates specific components of the graphics support system which are being

specified by the task force. The primary components are visual objects (in particular, their definition and manipulation), window management, and image generation system.

An object-oriented approach is adopted in which the application program deals with high-level "visual objects", such as menus and icons, rather than just primitives as points and lines.

3.7.1 Visual Objects

The visual objects will deal with the specification and coordination of encapsulated graphical entities. These visual objects provide a higher level of support to the applications developer than a subroutine library consisting of graphics primitives and control over viewing transformations. Visual objects can provide an organized tool kit of objects with a powerful set of properties and behaviors.

The visual objects prototype consists of several identifiable functions and interfaces whose descriptions follow.

3.7.1.1 Subtask - Visual Objects Definition

The application program interface to the graphics toolset is based on an object-oriented approach to user interface design and implementation.

The goal of this function is to provide the framework for defining and manipulating visual objects. This involves classifying visual objects, describing their behavior and visual aspects, and indicating the relationships between them. This will be accomplished through the development of an object definition language

Visual objects fall into two categories: those commonly used by application programmers in developing user interfaces (e.g., frames, menus and icons), and those used in specific application domains (e.g., markers representing military units). This function provides for the identification and definition of these visual objects using the framework described above.

3.7.1.2 Visual Representation

Objects may be represented visually in a number of ways, such as shaded images or wireframes. This function provides the link between the abstract concept of a visual object and its visual representation

In this function the forms, mechanisms, and the techniques for storing, retrieving, and defining visual representations, such as in a symbol library or database will be designated.

3.7.1.3 Visual Objects Coordination

A mechanism for coordinating the behavior of visual objects is necessary. This may be accomplished using one or more higher level objects which are responsible for distributing messages to the visual objects from other system components, such as an application program or the window manager. An object display list manager is an example of a higher level coordinator.

A command processor will be specified to exert some control over the behavior of visual objects. Such a command processor might be associated with an application or one of the other graphics modules, such as the input switcher. This function will specify how the coordination of visual objects should be distributed and shared.

3.7.1.4 Application Interface

T he application interface describes the common support capabilities provided by the visual objects. This defines the protocol between the application developer and the object-oriented Ada packages that applies to every object.

In addition to the set of commands of an object that are invoked directly by the user, there is also a set that is accessed indirectly through higher level objects responsible for distributing the messages to them. These higher level objects coordinate the display and behavior of the visual objects. They rely on support from the common protocol that each visual object must define.

3.7.1.5 Application Database Interface

The visual objects may need to respond to changes in the application database. Certain items in the database represent the state of the application. The user must be able to continuously monitor this state. When one of the state items changes, the visual object must be informed so it can change its image on the screen. These changes might include the color, size, position, or orientation of the object.

In order to support dynamic responses to database changes, there must be an interface between the application database and the visual objects. This interface lies between the graphics task group and the database task group. In the graphics task group we will specify the functions supported by this interface. The database task group will evaluate the task of defining the detailed syntax and semantics of the interface.

3.7.1.6 Image Generation Package Interface

The purpose of this subtask is to define the generic functionality of the image generation packages used by the visual objects. It is anticipated that several image generation packages

will be used in the graphics system. These packages include the Programmer's Hierarchical Interactive Graphics Standard (PHIGS), the Graphics Kernel System (GKS), and Computer Graphics Interface (CGI). Since the objects may be dealing with several different image generation packages, there is a need to define a minimal functionality for these packages. This functionality can be divided into two parts, output and input.

On the output side, there may be features of the image generation package that may be required by all or a large range of visual objects. These features include geometric primitives, transformations, and visibility control. It is not the intention of this project to define the syntax and semantics of these features, but to ensure that they are presented. On the input side, there must be some way of converting the input information coming from the image generation package into messages for the visual objects.

3.7.2 Window Manager

The objective of this task is to define a window management facility and its interface to the rest of the support system. The window manager will be consistent with other graphics support components and be appropriately bound to the kernel operating environment.

The window manager manages the screen real estate of the display devices and provides the device and application independent interface to the display surfaces. The window manager responds to commands from the application program to create, site, and position windows, and may also notify programs of changes to the window world. It provides the image generation packages with "virtual screens" for generating graphics output, and also provides clipping services and input demultiplexing. Since there is a single window manager view provided to all applications and image generation packages, performance and non-blocking behavior are critical.

3.7.3 Image Generation

This task group is primarily responsible for the content and requirements of image generation packages using existing or soon to be graphics standards.

An objective is to create a PHIGS binding to Ada with the purpose of analyzing the compatibility between the PHIGS semantic and the Ada semantic. Such a binding should be usable as a means of expressing the functionality by the visual objects group in terms of PHIGS functionality. Any incompatibilities should be raised as issues requiring further resolution.

This task will also specify the interfaces for the PHIGS input facilities and specify the requirements of PHIGS in relation to its demands on the other tools of the system. Such tools

include CGI-level graphics functionality and window manager support

3.7.4 CGI

An objective of this group is to analyze the functionality of the CGI within the context of the WIS graphics conceptual model. Particular concerns are its input facilities and support of faster operations. Interfaces to other graphics components, such as the Image Generation Package (IGP), the window manager, the window clipper, and the input switcher, need to be defined. Issues to be resolved include the role of CGI in a windowing environment and its relationship to the IGP (e.g.,PHIGS). For example, should the window manager interface directly with the CGI, or indirectly via a minimal IGP? Should visual objects be able to directly invoke the CGI? Should the window clipper be above the CGI, or below it?

A second objective of the group is to create a CGI binding to Ada.

3.7.5 Graphics and Input Command Processing

This task is to specify the handling of input events from the keyboard, mice, or other devices. This is usually controlled by a centralized module called the Input Command Processor (ICP). The ICP imposes a consistent style of interaction and assists application programs in specifying their user interfaces.

A central issue to be resolved and specified is the interface between the graphics support modules and the command processor modules. With the visual object model, who controls the behavior of visual objects: the objects themselves or an external ICP? Other issues concern the use of a Command Processor (CP) in handling window manager commands, the nature of the input switcher, and the use of one or more listener windows for multiple input devices.

3.7.6 Status

The preliminary specifications are being analyzed and reformulated. Two additional iterations are anticipated before final specifications are ready which is due in the July 1985 and October 1985 timeframe.

3.8 Network Protocols

The performance of WIS operational functions requires the ability of WIS users to transfer a variety of types of information to both local and remote users and systems. Within a site, access to WIS capabilites, both the functions and data, will be accomplished using a LAN architecture. Access to the other sites and remote WIS systems and data will be accomplished using WIS

intercomputer networking capabilities through the Defense Data Network (DDN).

The requirements for network services necessitate tools to integrate the communications design for WIS. To support the distributed processing concept, any number of components (hardware and software) from different sources must be able to communicate among themselves through the use of predetermined protocols. The WIS will be built upon the current Department of Defense (DoD) protocols. However, activity in the international standardization community indicates that at some time in the future International Standard Organization (ISO) protocols may be used for the network communication functions. As the services required within the WIS are expanded the need for new applications and lower levels of protocols and services will grow. Based upon the experience of the ARPANET, methods which can reduce the complexity and resources needed for new protocol specification and implementation will be necessary. The use of the DDN as the long haul backbone for WIS raises the questions of how inter-LAN communications routing will be accomplished when additional factors beyond shortest distance must be accommodated.

The Task Force for Network Protocols has been established to investigate and identify those technical issues which will require prototyping to understand and implement solutions to networking problems. The prototype projects initially identified by the Task Force are those which it believes will yield both near and far term payoff for the WIS Joint Program Management Office (JPMO).

3.8.1 Common Ada Implementation of Open System Interconnection (OSI) and DoD Transport and Internet Protocols

3.8.1.1 Terminology

In this project description, TCP/IP refers to the combination of the DOD transmission control protocol and internet protocol. TP-4/CLIP refers to the combination of the ISO transport protocol (class 4) and the ISO connectionless internet protocol (CLIP),currently being defined as a subprotocol of the network layer. For brevity, TCP/IP and TP-4/CLIP are often referred to simply as "the DoD" and "the ISO" protocols, respectively.

3.8.1.2 Objectives of the Project

This project has several related general and specific objectives. The general objectives are as follows:

a. To provide technical input to the decision-making process regarding adoption of the ISO protocols by DoD.

b. To provide a tested, technical approach for making the transition to the use of the ISO protocols by DoD which will help reduce the risk and cost of such a move.

c. To contribute to the technical capability of DoD networks to interwork with ISO networks.

Given that a common design appears feasible, the specific objectives are as follows :

a. To investigate design issues associated with the following requirements:

(1) A common Ada interface is required for implementations of the DoD and ISO protocols, which would enable higher level software to use either of the protocols in as flexible a manner as possible, thereby paving the way for a low-cost, low-risk transition to the use of the ISO protocols.

(2) A common Ada software organization is required for the DoD and ISO protocols, making full use of Ada's packaging capabilities. The extent to which this is practically possible will provide a measure of the confidence which can be placed in the commonality of function of the two protocols.

(3) It is expected that the Ada packages emerging from the satisfaction of this objective will be useful for the implementation of gateways between DoD and ISO networks. The investigation will contrast the common organizational approach with that of providing only a common user interface to otherwise completely independent Ada implementations of the two protocols.

(4) It should be possible to use the packages in the common design in a transport level gateway between DoD and ISO networks.

b. To provide high-level designs reflecting the results of the above analysis.

c. To provide implementations and demonstrations of the designs.

3.8.1.3 Status of the Project

The Task Force for Network Protocols is actively engaged in the preparation of the final specification of this project. The final specification will be completed by 15 July 1985.

3.8.2 Automatic Generation of Ada Protocol Software for WIS

3.8.2.1 Introduction

The WIS will require new communications protocols over its life cycle. Traditionally, protocol software has required an extended period of time for development. The processes of formal specification, and implementation have been accomplished with techniques which necessitate mapping from one conceptual framework to another. Considerable effort has been expended to verify that the protocol requirements have been accurately mapped into the software solution space.

The Ada software methodology offers the protocol designer a powerful approach to overcome the requirements conversion problem and to achieve

extended period of time for development. The processes of formal specification, and implementation have been accomplished with techniques which necessitate mapping from one conceptual framework to another. Considerable effort has been expended to verify that the protocol requirements have been accurately mapped into the software solution space.

The Ada software methodology offers the protocol designer a powerful approach to overcome the requirements conversion problem and to achieve modularity, uniformity, completeness, and confirmability. Building upon the facilities inherent in Ada the automated generation of protocol software directly from the functional specification will result in a considerable savings.

Some limited success with automated production of protocol software from formal protocol specifications has been achieved so far and the field offers the prospect of a high level of success by the 1990's.

The successes with automatic generation to date have been achieved when protocol specification is done with state machine or equivalent methods, such as petri nets. Both language and graphical forms have been used for the specifications. Automatic code generation may be performed by translating individual state machine specifications into program fragments (usually subprograms) in languages such as Pascal or C. However these subprograms do not by themselves implement the protocols. Human intervention is required to devise a control framework appropriate for a particular hardware and operating system environment and to integrate the automatically generated procedures into this framework. Further human intervention is required to validate the results.

3.8.2.2 Objectives of the Project

This project will take an initial step towards the automatic generation of Ada protocol software for WIS. The initial step is in two parts:

a. Implementation of a prototype code generator for Ada code fragments following work already done in the protocol field for other languages

b. Analysis of the feasibility of developing more automatic methods which not only produce code fragments but also provide means of integrating them into an appropriate control framework for an implementation

The ultimate objective for the 1990's is to produce Ada protocol software from a combination of protocol specifications and additional system design information, such that only minimal hand tailoring of the resulting Ada programs will be required.

Two major streams of research activity world-wide affect this project: formal techniques for protocols and embedded systems design. Techniques from both of these areas will be needed to solve the general problem.

a. Formal techniques for protocols: This activity includes formal specifications for communication protocols and their translation into implementations.

b. Embedded systems design: This activity includes operational software design methods and rapid prototyping methods for event-driven, realtime embedded systems.

This project will specifically address the following:

a. What specification technique should be used for the protocols? Techniques in current use include, both separately and in combination, narrative text, timing diagrams, coupled state machines, petri nets, temporal logic and special languages.

b. What techniques should be used to determine the implementation framework (the logical organization of the implementation), based on the nature of the protocols to be implemented? Currently there are no automatic techniques available for doing this; it is a matter for human design judgement.

c. How and at what stage of the process should the internal protocols be specified? Because these are protocols between modules of an implementation, some decisions must be made about the implementation framework first.

d. How can the protocol specifications be used to generate Ada implementation code?

e. How can the code fragments generated from the protocol specifications be integrated into a particular implementation framework?

f. How can the results be validated?

3.8.2.3 Status of the Project

The Task Force for Network Protocols is actively engaged in the preparation of the specification for this project. The final specification will be complete by 15 July 1985.

3.8.3 Multi-Varible Objective Functions as a Basis for Network Routing in WIS

3.8.3.1 Introduction

WIS is composed of a moderate number (less than 50) of LAN's. The LAN's will service a heterogeneous mix of terminals and hosts. The DDN will be used as the long haul backbone. WIS inter-operability is required with other networks, domestic and foreign, commercial and military. WIS requirements that are unique and/or stringent include multi-level security, priority, reliability, reconfigurability, and distributed databases. Applications to be serviced include teleconferencing, Telnet, file/data transfer, wide band voice and data, narrow band voice, video, and facsimile.

The DDN is based on a datagram service similar to, if not identical with, the current Defense Advanced Research Projects Agency (DARPA) set of protocols. The nature of the WIS LAN protocols is not clear at this time; however, all WIS LAN's will be connected via multiple gateways to the DDN. The LAN-to-DDN gateways will be responsible for routing, other network services, and providing additional services required by applications of WIS that are not currently provided by standard internet protocols.

The DDN will provide connections between all gateways, routing procedures in the WIS gateways and DDN gateways will determine the "best" communications paths. The criteria used by DDN will be its own, e.g. shortest hop or minimum delay. However, these criteria are those which have been developed for networks with a limited number of decision varibles governing the choice of a "best" communications routing. The more complex nature of the WIS, compared to the WWMCCS or ARPANET, dictates that the "best" communications routing be selected based upon an integrated consideration of all criteria. Some of these new criteria are priority, survivability in a dynamic environment, and reliability for multiple classes of service.

The final objective of this project is the production of prototype communications routing algorithms which will encorporate multiple decision variables. Since the WIS and DDN gateways will have to share the routing responsibilities, it is anticipated that the results of this project will become a foundation for improved WIS and DDN communications routing.

3.8.3.2 Objectives of the Project

Several aspects of inter-LAN communications need to be addressed before an inter-LAN mechanism can be implemented. First, a good routing protocol must be developed. Second, the gateway must be designed for high performance. The routing protocol itself comprises two components: an algorithm for computing the routes based on some cost function, and the protocol which uses this route information to perform the inter-LAN communications. We envision a multi-year effort on all parts of inter-LAN communications which will result in an Ada implementation of these mechanisms. Efforts should be undertaken in the areas of algorithms, routing, and implementation.

The objective of the work with algorithms is to develop efficient routing algorithms that will provide "good" routes within WIS under varying traffic loads and topological changes. For the purpose of this study, we assume that WIS consists of n LAN's. Each LAN is connected to DDN through at least two gateways. Each gateway is connected to DDN by one or more links. All connections from a LAN to DDN are to different parts of DDN. Simplified assumptions regarding the workload will be made.

a. A simple problem will be investigated initially where the routing algorithm is concerned with optimizing a single objective function, say ND[1], with no constraints. This problem has been studied previously, and several approaches have appeared in the literature.

b. A problem where the routing algorithm is concerned with optimizing an objective function that contains two or more terms weighted together. An example is a[1]*ND[1]+a[2]*ND[2]. This problem has no constraints.

c. This problem contains a single objective function which may be a weighted sum of several performance measures, and one or more constraints on the remaining performance measures. An example of this problem is

    minimize ND[1]

    subject to NR[2] > r.

d. This problem contains no objective function but merely requires a set of routes such that a set of constraints are satisfied.

e. This problem asks for a set of routes that optimize several objective functions.

Work in the routing area will be concerned with routing for systems with non-convex objective functions and many classes of traffic. Priorities would be included at this time as they appear to introduce non-convexity. The most promising approach or approaches identified during the earlier phase of study would be considered at this time. The result of this phase of study will be the routing algorithm to be used in WIS.

Regarding implementation, the algorithms developed during the first phases of study would be implemented in Ada and evaluated on a testbed system.

3.8.3.3 Approaches to Problem

There appear to be two approaches to solving the above problems decentralized (distributed) and centralized. In the first case, all gateways take part in executing the algorithm.

Moreover, each gateway uses a portion of all the information required to execute the algorithm. A centralized algorithm requires that all necessary information be collected at one processor. The algorithm is then executed at that processor using this information. The centralized approach can also be implemented whereby all information is collected at all gateways and each gateway executes the same algorithm. This second approach is currently used by ARPANET. There are several issues involving each of these approaches that require study.

The decentralized approach produces algorithms that require several iterations before the optimal routes are produced. There is a question regarding how many steps are necessary for convergence. There is also a question regarding the length of time necessary to perform a single step. The answers to both of these questions will determine how well decentralized algorithms can handle changes in workload and in topology.

3.8.3.4 Status of the Project

The Task Force for Network Protocols is actively engaged in the preparation of the specification of this project. The final specification will be completed by 15 July 1985.

3.8.4 Status and Plans

We expect to complete detailed specification and statements of work in each of the above network protocol areas by September 1985.

APPENDIX A

LIST OF CONSULTANTS

Appendix A contains a list of consultants who have contributed to this document.

SOFTWARE DESIGN DESCRIPTION AND ANALYSIS TOOLS TASK FORCE

IDA Task Interface: Bob Knapper

Task Force Leader:

Leon Stucki
724 West Hi-Crest Dr.
Auburn, WA 98001
(206) 939-7552
lstucki@@eclb

Chair Emeritus:

William E. Riddle
Software Design and Analysis, Inc.
1670 Bear Mountain Dr.
Boulder, CO 80303
(303) 499-4782
riddle@@usc-eclb

Task Force Members:

Betsy Bailey - Software Metrics
400 N. Cherry St.
Falls Church, VA 22046
(703) 241-3949
bbaily@@eclb

Lee Osterweil - Feedback Analysis
Chair, Computer Science Department
University of Colorado
Boulder, CO 80309
(303) 492-8787
ida%boulder.csnet@csnet-relay

Christine Youngblut - PDL
Computer Technology Associates
17021 Sioux Lane
Gaithersburg, MD 20878
(301) 948-1989
cyoungblut@@eclb

Grady Booch
835 S. Moore St.
Lakewood, CO 80226
(303) 986-2405
gbooch@@eclb

TEXT PROCESSING

IDA Task Interface: Murray Berkowitz

Task Force Leader:

Alan C. Shaw
University of Washington
Department of Computer Science
Seattle, WA 98195
(206) 543-9298
shaw@@washington

Task Force Members:

Christopher W. Fraser
Computer Science Department
University of Arizona
Tucson, AZ 85721
cwf@@arizona.csnet@csnet-relay

Ned Irons
Slater Towers, Ltd.
586 Longs Peak Road
Longs Peak Route
Estes Park, CO 80517
(303) 586-5933

Other Portential Members:

Ben Schneiderman (University of Maryland)

COMMAND LANGUAGE

IDA Task Interface: Joesph Hrycyszyn

Task Force Leader:

Dr. Thomas S. Kaczmarek
USC/ISI
4676 Admiralty Way
Marina del Ray, CA 90292
(213) 822-1511, ext. 271
kaczmarek@@isif

Task Force Members:

Philip J. Hayes
Carnegie-Mellon University
Computer Science Department
Pittsburgh, PA 15213
(412) 578-2631
phil.hayes@@cmu-cs-a

Robert J. K. Jacob
Naval Research Laboratory
Code 7590
Washington, D.C. 20375
(202) 767-3365
jacob@@nrl-css

Jon A. Meads
User Interaction Computer Graphics
Software and Computer Systems
2516 NE 19th Street
Portland, OR 97212
jmeads@@eclb

Ken R. Holmes
M/A-COM LINKABIT, Inc.
3033 Science Park Rd.
San Diego, CA 92121
(619) 457-2340

DATABASE TOOLS

IDA Task Interface: Murray Berkowitz

Task Force Leader:

Gio Wiederhold
Computer Science Department
Stanford University
Palo Alto, CA 94305
(415) 497-0785
wiederhold@@sumex-aim

Task Force Members:

Arthur M. Keller
Stanford University
Computer Science Department
Palo Alto, CA 94305
(415) 497-3227
ark@@su-ai

David L. Spooner
Department of Mathematical Science
Rensselaer Polytechnic Institute
Troy, NY 12181
(518) 266-6890
spooner%rpi.csnet@@csnet-relay

OPERATING SYSTEMS

IDA Task Interfaces: Clyde Roby and John Salasin

Task Force Leader:

John (Jack) Stankovic
University of Massachusetts
Electrical and Computer Engineering Department
Amherst, MA 01003
(413) 545-0720
stankovic%umass-ece.csnet@@csnet-relay

Task Force Members:

Ming T. Liu
Department of Computer and Information Science
The Ohio State University
2036 Neil Avenue Mall
Columbus, OH 43210
(614) 422-1837
liu@ohio-state.csnet@@csnet-relay

David Cheriton
Computer Science Department
Stanford University
Stanford, CA 94305
(414) 497-1054
cheriton@@pescadero

Alan J. Smith
Department of Electrical Engineering and Computer Science
531 Evans Hall
University of California
Berkeley, CA 94720
(415) 642-5290
smith@@ucb-vax
Home office:
7026 Norfold Rd.
Berkeley, CA 94705
(415) 549-3190

PLANNING AND OPTIMIZATION AIDS

IDA Task Force Interface: Mike Bloom

Task Force Leader:

Dr. Alexander Levis
Laboratory for Information and Decision Sciences (LIDS)
Massachusetts Institute of Technology
Cambridge, MA 02139
(617) 253-7262
levis@@mit-multics

Chair Emeritus:

Harold Sorenson
Chief Scientist
United States Air Force
HQ USAF/CCN
The Pentagon
Washington, D.C. 20330-5040
(202) 697-7842
sdcc6!is81@@ucsd
Home office:
5500 Teak Court
Alexandria, VA 22309

Task Force Members:

Dr. Vivek S. Samant
2117 Brigantine Ct.
Encinitas, CA 92024
(619) 436-7317
vsamant@@eclb

Professor Stephen J. Kahne
Department of Electrical Engineering
Polytechnic Institute of New York
Brooklyn, NY 11201
(212) 330-0362 (home)
skahne@@eclb

William Kilmer
Department of Electrical and Computer Engineering
University of Massachusetts
Amherst, MA 01003
(413) 545-0672 (office)
(413) 545-2442 (messages)
bkilmer@@eclb

Andrew P. Sage
First American Professor of Information Technolgy
George Mason University
4400 University Drive
Fairfax, VA 22030
asage@@eclb

GRAPHICS

IDA Task Interface: Joe Hrycyszyn

Task Force Leader:

James Foley
Computer Graphics Consultants, Inc.
510 H Street, S.W.
Washington, D.C. 20024
(202) 554-8021
jfoley@@eclb
(Jim Templeman and Patti Denbrook are working with J. Foley.)

Task Force Members:

Dr. Mark Weiser
Computer Sciences Department
Room 4315
University of Maryland
College Park, Maryland 20742
(301) 454-7817
mark@@maryland

Dr. Mark Green
Department of Computer Science
University of Alberta
Edmonton, Alberta, Canada 26G2HI

Brad Myers
2085 Islington Av.
Suite 610
Eston, Ontario, Canada M9P3R1
(416) 978-6986 (office)
(416) 244-4083 (home)
bmyers@@eclb

Kathleen A. Gilroy
Software Productivity Solutations, Inc.
P.O. Box 361697
Melbourne, FL 32936
(305) 254-4268
kgilroy@@eclb

Dr. Richard Puk
Puk Consulting Services
7644 Cortina Ct.
Carlsbad, CA 92008
(619) 753-9027
rpuk@@eclb

Jeffrey E. Simon
M/A-COM LINKABIT, Inc.
3033 Science Park Rd.
San Diego, CA 92121
(619) 457-2340 (office)
(619) 755-6253 (home)
jsimon@@eclb

NETWORK PROTOCOLS

IDA Task Interface: Mike Bloom

Task Force Members:

Robert Boorstyn
Polytechnic Institute of New York
333 Jay St.
New York, NY 11201
(212) 643-4485
boorstyn@@isi

Ray Buhr
Department of Computer and Systems Engineering
Carleton University
Ottawa, Ontario, Canada K1S5B6
(613) 231-2645
buhr%carleton.cdn%ubc.csnet@@csnet-relay

Don Towsley
Department of Electrical and Computer Engineering
University of Massachusetts
Amherst, MA 01003
(413) 545-0766
towsley%umass-ece.csnet@@csnet-relay

Dennis MacKinnon
15 Pansy Av.
Ottawa, Ontario, Canada KIS2W5
(613) 234-1003

APPENDIX B

LIST OF TERMS AND ACRONYMS

Appendix B contains a list of the terms and abbreviations contained within this document.

ANSI	American National Standards Institute
APSE	Ada Programming Support Environment
CAD	Computer-Aided Design
CAIS	Common Ada Programming Support Environment (APSE) Interface Set
CGI	Computer Graphics Interface
CL	Command Language
CLIP	Connectionless Internet Protocol
CODASYL	Conference on Data System Languages
COSCL	Common Operating System Command Language
COTS	Commercial Off-the-Shelf
CP	Command Processor
DARPA	Defense Advanced Research Projects Agency
DBM	Database Machine
DBMS	Database Management System
DDL	Data Definition Language
DDN	Defense Data Network
DEC	Digital Equipment Corporation
DoD	Department of Defense
DSS	Deterministic Sequencing and Scheduling
DGMS	Dialog Generation and Management System
dpANS	Draft Proposal American National Standard
FY	Fiscal Year
GKS	Graphics Kernel System

ICP	Input Command Processor
IGP	Image Generation Package
IP	Internet Protocol
IPC	Inter-Process Communication
ISO	International Standards Organization
JOPES	Joint Operations Planning and Employment System
JPMO	Joint Program Management Office
KP	Knapsack Problem
LAN	Local Area Network
LRM	Language Reference Manual
MBMS	Model-Based Management System
MIP	Mixed Integer Linear Programming
NOS	Network Operation System
NOSC	Naval Ocean Systems Center
NSA	National Security Agency
OSCRL	Operating System Command and Response Language
OSI	Open System Interconnection
PDL	Program Design Language
PHIGS	The Programmer's Hierarchical Interactive Graphics Standard
QAP	Quadratic Assignment Problem
SQL	Structured Query Language
SSA	Sequencing, Scheduling, and Assignment
TCB	Trusted Computing Base
TCP	Transmission Control Protocol
TP	Transport Protocol
UNITREP	Unit Status and Reporting
WAN	Wide Area Network

WIS	WWMCCS Information System
WISSAPED	WIS Smart Advisor for Planning and Execution of Decisions
WOS	WIS Operating System
WWMCCS	World Wide Military Command and Control System
WYSIWYG	What You See Is What You Get

Distribution List for IDA Paper P-1869

NAME AND ADDRESS	NUMBER OF COPIES
Sponsor	
CPT Stephen Myatt WIS JPMO/DXP Room 5B19, The Pentagon Washington, D.C. 20330-6600	5
Other	
Defense Technical Information Center Cameron Station Alexandria, VA 22314	2
Mr. Bill Allen 311 Park Place Blvd. Suite 360 Clearwater, FL 34619	1
Ms. Linn Roller General Dynamics P.O. Box 748 M-2 748 Ft. Worth, TX 76101	1
Dr. John Salasin GTE 1700 Research Blvd. Rockville, MD 20850	2
Mr Eugen Vasilescu 35 Chestnut St. Malverne, Long Island, NY 11565	1
CSED Review Panel	
Dr. Dan Alpert, Director Program in Science, Technology & Society University of Illinois Room 201 912-1/2 West Illinois Street Urbana, Illinois 61801	1

NAME AND ADDRESS	NUMBER OF COPIES
Dr. Barry W. Boehm TRW Defense Systems Group MS R2-1094 One Space Park Redondo Beach, CA 90278	1
Dr. Ruth Davis The Pymatuning Group, Inc. 2000 N. 15th Street, Suite 707 Arlington, VA 22201	1
Dr. C.E. Hutchinson, Dean Thayer School of Engineering Dartmouth College Hanover, NH 03755	1
Mr. A.J. Jordano Manager, Systems & Software Engineering Headquarters Federal Systems Division 6600 Rockledge Dr. Bethesda, MD 20817	1
Mr. Robert K. Lehto Mainstay 302 Mill St. Occoquan, VA 22125	1
Dr. John M. Palms, Vice President Academic Affairs & Professor of Physics Emory University Atlanta, GA 30322	1
Mr. Oliver Selfridge 45 Percy Road Lexington, MA 02173	1
Mr. Keith Uncapher University of Southern California Olin Hall 330A University Park Los Angeles, CA 90089-1454	1
IDA	
General W.Y. Smith, HQ	1

NAME AND ADDRESS	NUMBER OF COPIES
Mr. Philip Major, HQ	1
Dr. Robert E. Roberts, HQ	1
Mr. Michael I. Bloom, CSED	1
Mr. Bill R. Brykczynski, CSED	5
Ms. Anne Douville, CSED	1
Dr. John F. Kramer, CSED	1
Mr. Terry Mayfield, CSED	1
Ms. Katydean Price, CSED	2
IDA Control & Distribution Vault	3